Predicting and Controlling Resource Demand in Heterogeneous Active Networks

Virginie Galtier	Université Henri Poincaré
Kevin Mills	NIST
Yannick Carlinet	France Telecom
Stephen Bush	GE CR&D
Amit Kulkarni	GE CR&D

MILCOM 2001

October 30, 2001

Presentation in Essence

- **<u>Problem</u>: Growing use of mobile code among heterogeneous platforms increases the need to predict and control CPU usage, while simultaneously increasing the challenge of doing so.**

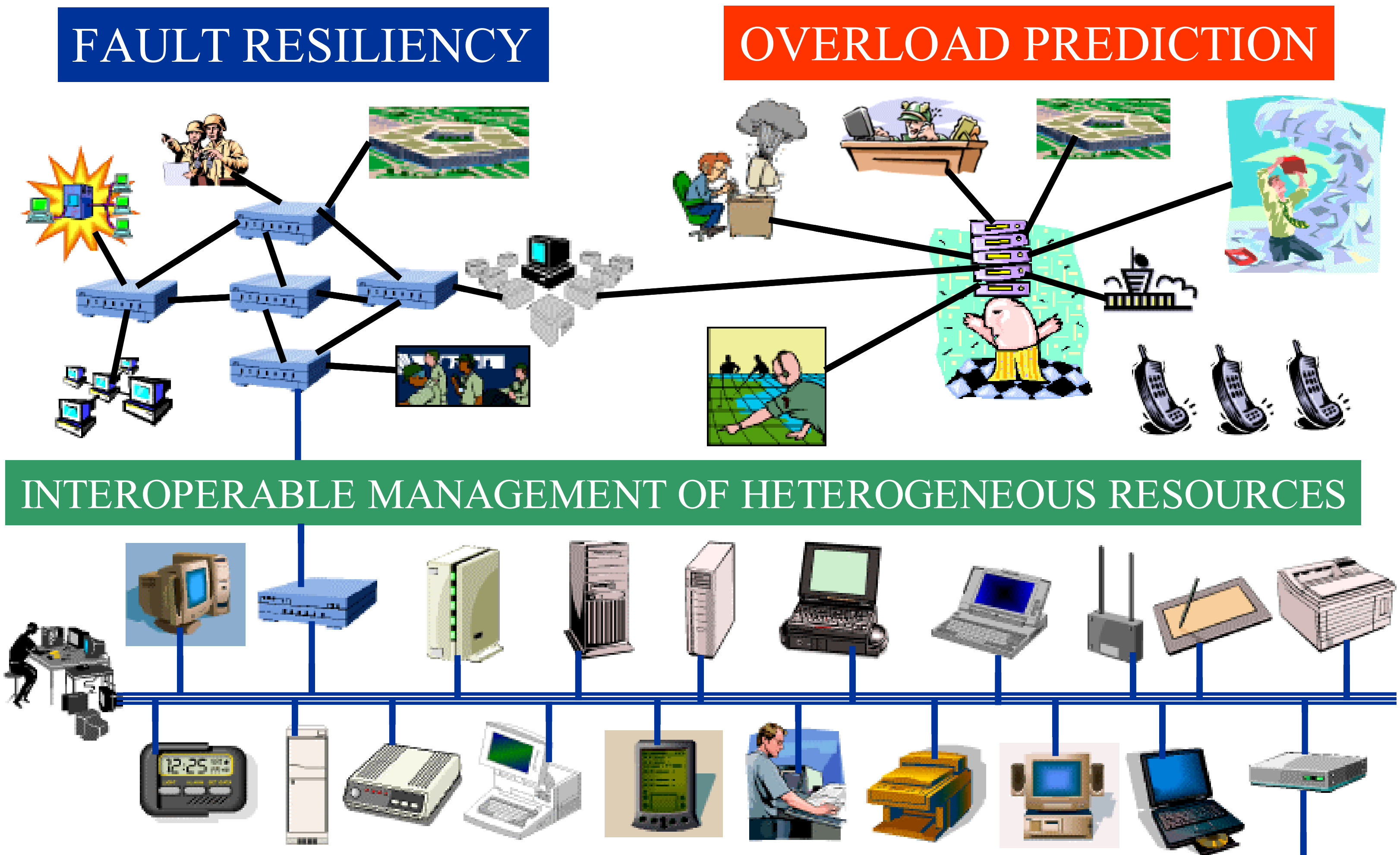
- **<u>Approach</u>: We devised a method to model CPU demands by mobile code distributed among heterogeneous nodes, and we evaluated our method when applied to predict and control CPU use in active networks, which represent an advanced application of mobile code.**

- **<u>Results</u>: Our method yielded improved performance in predicting CPU demand and enabled more precise control of CPU usage in a heterogeneous active network.** (Our MILCOM paper addresses only the prediction improvements.)

- **<u>Impact</u>: Many distributed applications rely increasingly on mobile code. Our work can help to improve resource estimation and control in such applications.** (But additional research is needed.)

(more information available at http://w3.antd.nist.gov/active-nets/)

The Problem


FAULT RESILIENCY
OVERLOAD PREDICTION
INTEROPERABLE MANAGEMENT OF HETEROGENEOUS RESOURCES
12:25

Growing Population of Mobile Programs on Heterogeneous Platforms

SCRIPTING ENGINES & LANGUAGES

tcl tk

Python

vbscript
jscript

APPLETS & SERVLETS

JAVA

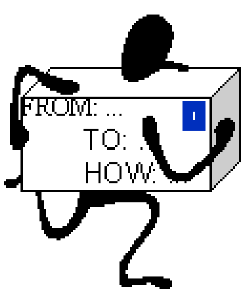
C#

dlls, dlls, and more dlls

MOBILE AGENTS

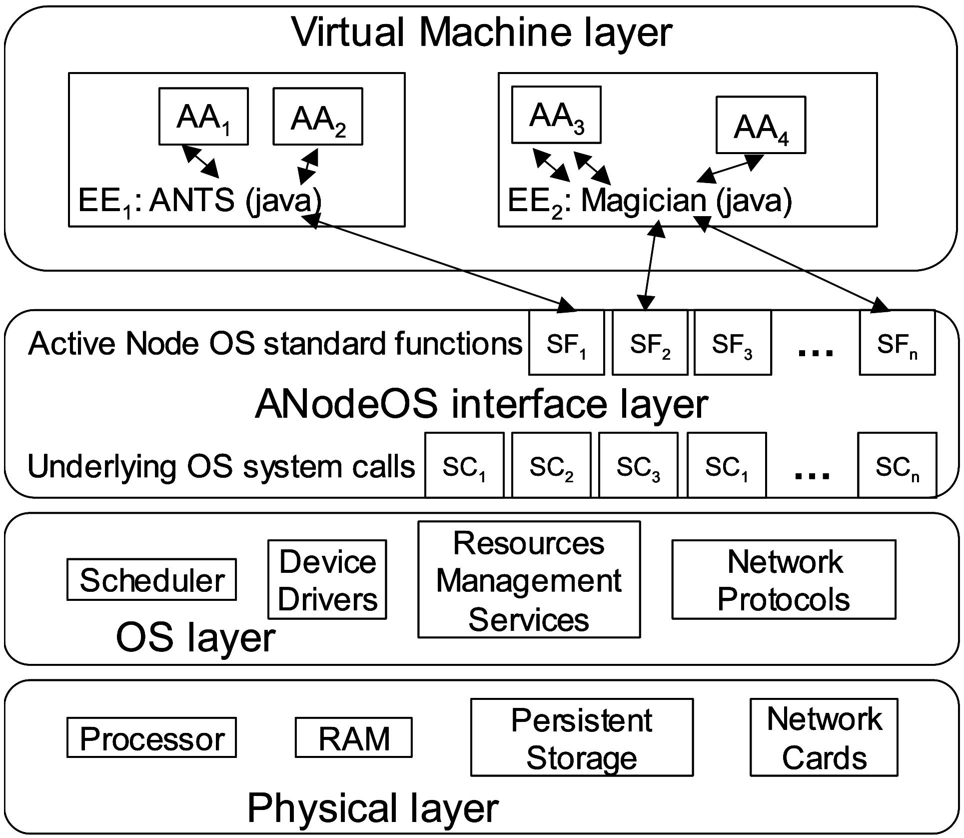
AgentSpace

D'Agents Dartmouth



Active Networks

Sources of Variability in Execution Environment and System Calls



ANETS ARCHITECTURE

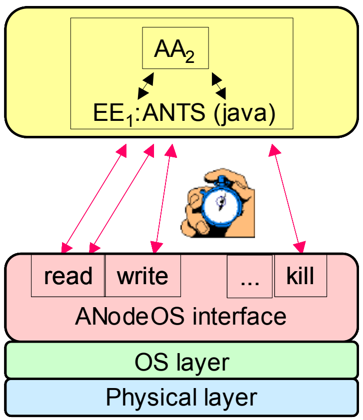
Trait	Blue	Black	Green
CPU Speed	450 MHz	333 MHz	199 MHz
Processor	Pentium II	Pentium II	PentiumPro
Memory	128 MB	128 MB	64 MB
OS	Linux 2.2.7	Linux 2.2.7	Linux 2.2.7
JVM	jdk 1.1.6	jdk 1.1.6	jdk 1.1.6
Benchmark			
Avg. CPU us	534	479	843
Avg. PCCs	240,269	159,412	167,830

	Blue		Black		Green	
System Call	pcc	us	pcc	us	pcc	us
read	19,321	43	12,362	37	12,606	63
write	22,609	50	14,394	43	12,362	62
socketcall	27,066	60	17,591	53	14,560	73
stat	22,800	51	14,731	44	12,042	61

Our Approach to Solve the Problem

Our Approach to Modeling CPU Demands

Monitor at System Calls in Active Node OS → **Generate Execution Trace** → **Generate Active Application Model** →



...
begin, user (4 cc), read (20 cc), user (18 cc), write(56 cc), user (5 cc), end

begin, user (2 cc), read (21 cc), user (18 cc), kill (6 cc), user (8 cc), end

begin, user (2 cc), read (15 cc), user (8 cc), kill (5 cc), user (9 cc), end

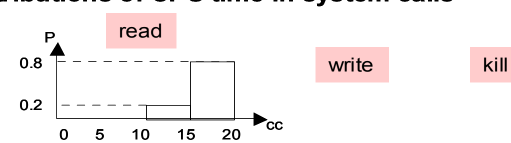
begin, user (5 cc), read (20 cc), user (18 cc), write(53 cc), user (5 cc), end

begin, user (2 cc), read (18 cc), user (17 cc), kill (20 cc), user (8 cc), end
...

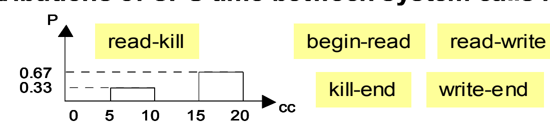
Trace is a series of system calls and transitions stamped with CPU time use

Scenario A:
sequence = “read-write”,
probability = 2/5
Scenario B:
sequence = “read-kill”,
probability = 3/5

Distributions of CPU time in system calls :



Distributions of CPU time between system calls :



→ **Simulate Model with Monte Carlo Experiment** → **Statistically Compare Simulation Results against Measured Data** →

		100 bins-20000 reps	
EE	AA	Mean	Avg. High Per.
ANTS	Ping	0.86	0.9
	Mcast	0.40	1.9
Magician	Ping	0.44	33
	Route	0.73	13

Scaling AA Models

AA model on node X:
```
read   30 cc
user   10 cc
write  20 cc
```

Model of node Y:
```
read   20 cc
write  45 cc
user    9 cc
```

scale

Model of node X:
```
read   40 cc
write  18 cc
user   13 cc
```

AA model on node Y:
```
read   30*20/40 = 15 cc
user   10*9/13  =  7 cc
write  20*45/18 = 50 cc
```

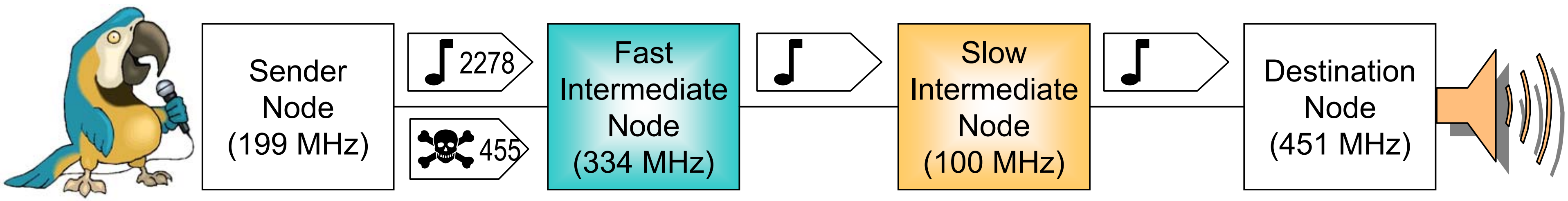
Our Model Predicts CPU Demands with Increased Accuracy

High percentiles aggregate 80th, 85th, 90th, 95th, and 99th percentiles

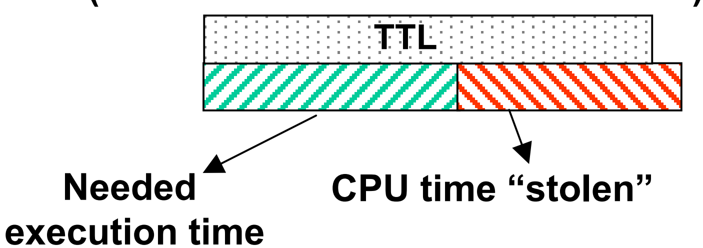
EE	AA	Node X	Node Y	Predictions after scaling with speed ratio: % Error mean prediction	Predictions after scaling with speed ratio: % Error high percentiles prediction	Predictions with NIST model: % Error mean prediction	Predictions with NIST model: % Error high percentiles prediction
ANTS	Ping	machine A	machine B	94	110	0.42	8
		machine D	machine C	31	19	-2	8
		machine E	machine C	23	29	-7	7
	Multicast	machine B	machine E	22	20	-2	12
		machine C	machine D	-11	11	-2	10
		machine A	machine C	226	209	5	9
Magician	SmartPing	machine E	machine C	34	30	-5	9
		machine B	machine C	121	103	-7	14
		machine A	machine D	287	281	-9	10
	SmartRoute	machine E	machine D	14	10	-2	24
		machine D	machine C	15	21	-5	9
		machine C	machine A	-81	81	-3	10

Evaluate Our Approach Applied to Control CPU Usage in a Heterogeneous Active Network

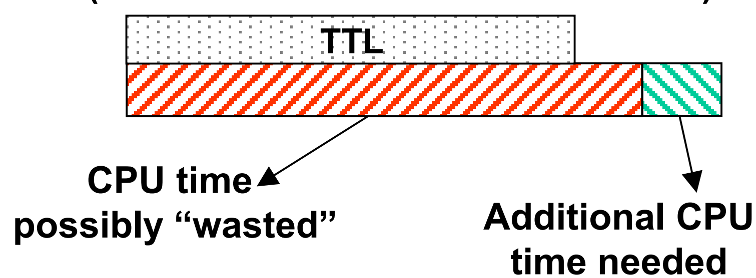
Goals: (1) Show reduced CPU usage by terminating malicious packets earlier ***AND***
(2) Show fewer terminations of good packets



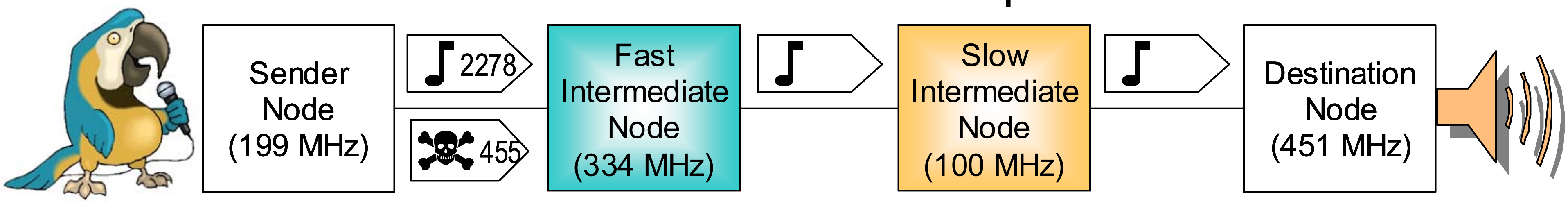
Malicious Packet dropped too late (CPU use reached TTL + tolerance)



Good packet dropped early (CPU use reached TTL + tolerance)



Results for CPU-Control Experiment



Control = Kill any packet that executes above predicted 99th percentile of execution time

	Sender	Fast	Slow
Measured:	**8.29 ms = 1,650,084 cc**	**4.76 ms = 1,589,382 cc**	**23.99 ms = 2,398,702 cc**

Experiment #1 predictions based on execution time on sender and processor speed ratio

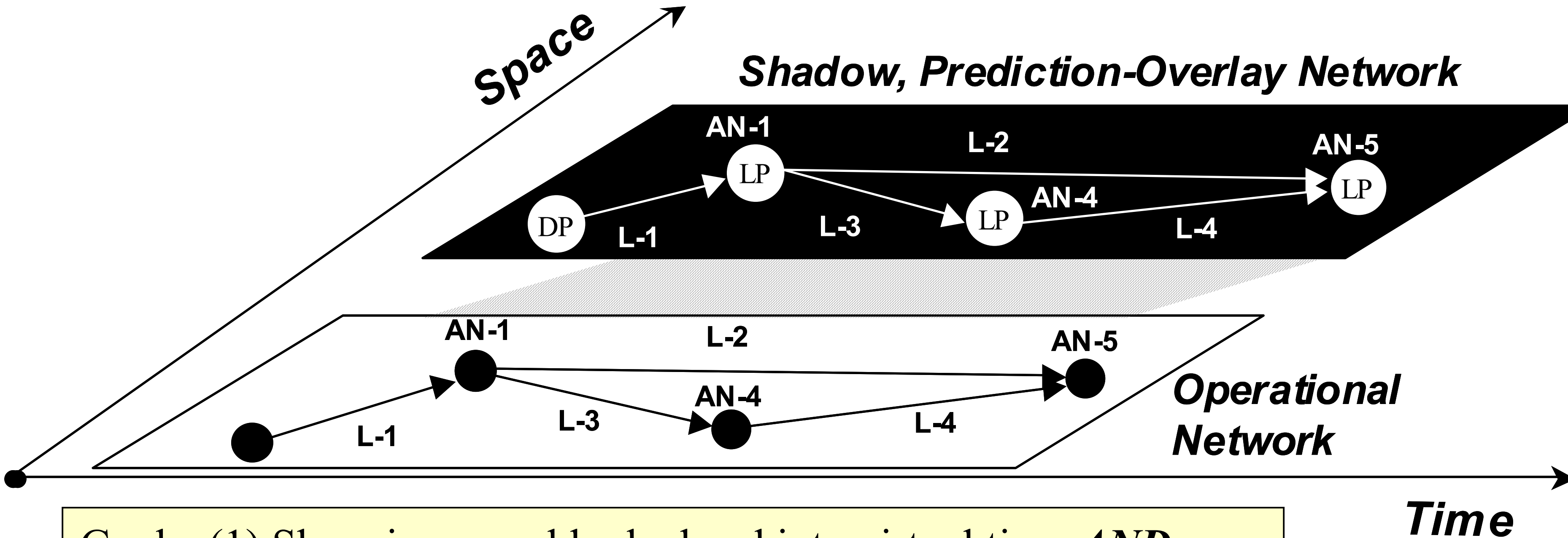
Fast	Slow
8.29 ms = 2,769,487 cc	**8.29 ms = 829,187 cc**
CPU Time Stolen **455 * 8.29 = 3,772 ms**	**2186 good packets are killed** **CPU Time Wasted = 18,122 ms**

Experiment #2: predictions obtained with NIST model

Fast	Slow
4.76 ms	**23.99 ms**
CPU Time Stolen **455 * (8.29 – 4.76) = 1,606 ms** **Improvement in avg. CPU use = 0.7 ms/packet**	**Only 19 good packets are killed** **CPU Time Wasted = 456 ms** **Improvement = 2167 packets saved!**

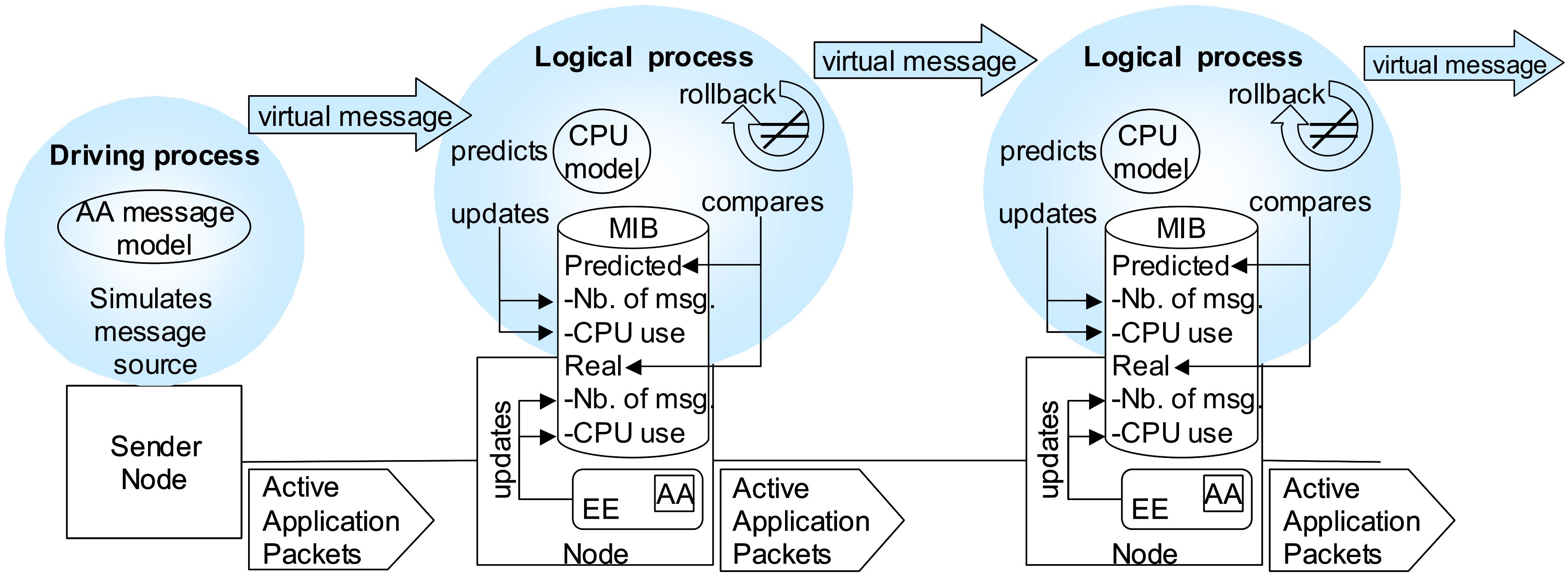
Evaluate Our Approach Applied to Predict CPU Demand in a Heterogeneous Active Network

Overlay network simulates application traffic ahead in virtual time.



Goals: (1) Show improved look ahead into virtual time ***AND***
(2) Show fewer tolerance rollbacks in the simulation

Active Virtual Network Management Prediction (AVNMP)

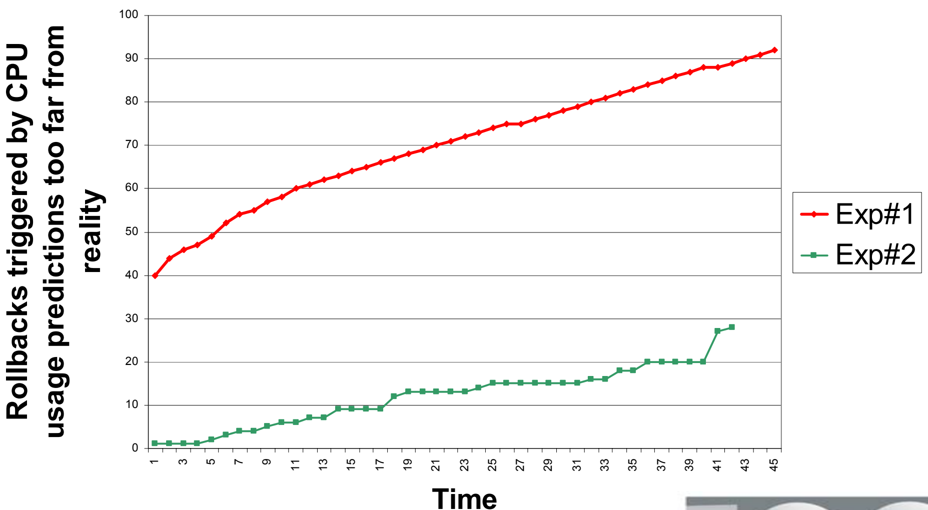


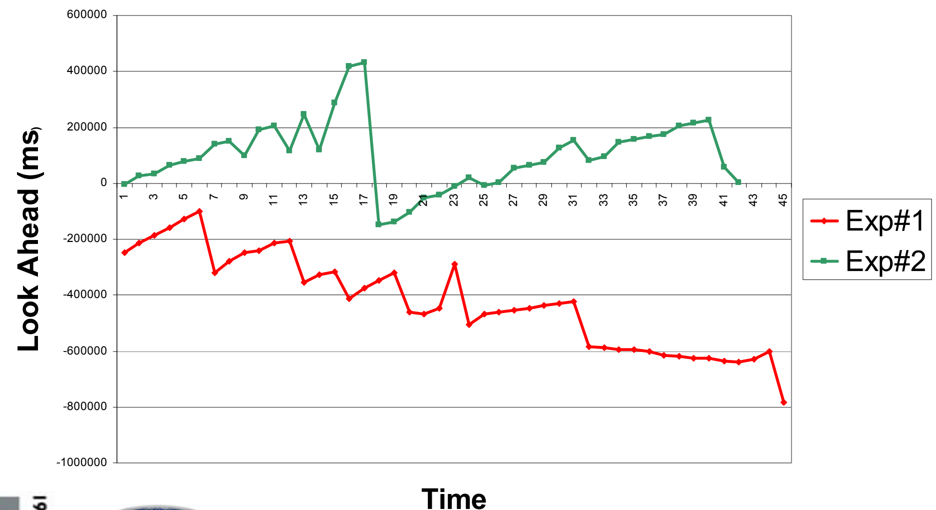
Experiment #1: CPU predictions based on average load on sender node and then transformed use processor-speed ratio
Experiment #2: CPU predictions obtained with NIST model
For both experiments: tolerance before rollback = 10 %.

Results for CPU-Prediction Experiments

	Exp#1: sender values scaled with processor speed ratio			Exp#2: CPU prediction with NIST model		
	first intermediate node	second intermediate node	destination node	first intermediate node	second intermediate node	destination node
maximum look ahead (seconds)	-101	-20	54	432	102	313
Rollbacks	92	42	12	28	0	0

AVNMP improvement on the first intermediate node:





Future Research

- Improve NIST Models
  - Space-Time Efficiency
  - Account for Node-Dependent Conditions
  - Characterize Error Bounds
- Investigate Alternate Models
  - White-box Model
  - Lower-Complexity Analytically Tractable Models
  - Models that Learn
- Improve AVNMP Performance

Presentation in Summary

- **Problem: Growing use of mobile code among heterogeneous platforms increases the need to predict and control CPU usage, while simultaneously increasing the challenge of doing so.**

- **Approach: We devised a method to model CPU demands by mobile code distributed among heterogeneous nodes, and we evaluated our method when applied to predict and control CPU use in active networks, which represent an advanced application of mobile code.**

- **Results: Our method yielded improved performance in predicting CPU demand and enabled more precise control of CPU usage in a heterogeneous active network.** (Our MILCOM paper addresses only the prediction improvements.)

- **Impact: Many distributed applications rely increasingly on mobile code. Our work can help to improve resource estimation and control in such applications.** (But additional research is needed.)

(more information available at http://w3.antd.nist.gov/active-nets/)